

An algebraic approach to enumerating non-equivalent double traces in graphs

Nino Bašić

Faculty of Mathematics and Physics, University of Ljubljana
Jadranska 19, 1000 Ljubljana, Slovenia
`nino.basic@fmf.uni-lj.si`

Drago Bokal

Faculty of Natural Sciences and Mathematics, University of Maribor
Koroška 160, 2000 Maribor, Slovenia
`bokal@uni-mb.si`

Tomas Boothby

Department of Mathematics, Simon Fraser University
Burnaby, B.C. V5A 1S6, Canada
`tboothby@sfu.ca`

Jernej Rus

IMFM
Jadranska 19, 1000 Ljubljana, Slovenia
`jernej.rus@gmail.com`

November 1, 2016

Abstract

Recently designed biomolecular approaches to build single chain polypeptide polyhedra as molecular origami nanostructures have risen high interest in various double traces of the underlying graphs of these polyhedra. Double traces are walks that traverse every edge of the graph twice, usually with some additional conditions on traversal direction and vertex neighborhood coverage. Given that double trace properties are intimately related to the efficiency of polypeptide polyhedron construction, enumerating all different possible double traces and analyzing their properties is an important step in the construction. In the paper, we study the automorphism group of double traces and present an algebraic approach to this problem, yielding a branch-and-bound algorithm.

Keywords: nanostructure design; self-assembling; topofold; polypeptide origami; double trace; strong trace; automorphism group of double trace; branch-and-bound.

Math. Subj. Class. (2010): 05C30, 05C45, 05C85, 68R10, 92E10.

1 Introduction

Gradišar et al. presented a novel self-assembly strategy for polypeptide nanostructure design in 2013 [14]. Their research was already improved by Kočar et al. in 2015, who developed another alternative strategy to design topofolds — nanostructures built from polypeptide arrays of interacting modules that define their topology [16]. Such approaches are paving the way to a significant breakthrough in the field of protein origami, an area advancing a step ahead of DNA origami, where many researchers have spent the better part of the past decade by folding the molecules into dozens of intricate shapes.

A polyhedron P that is composed from a single polymer chain can be naturally represented by a graph $G(P)$ of the polyhedron. As every edge of $G(P)$ corresponds to a coiled-coil dimer in the self-assembly process, exactly two biomolecular segments are associated with every edge of $G(P)$. Hence, every edge of $G(P)$ is in its biomolecular structure replaced by two copies, resulting in a graph $G'(P)$ obtained from $G(P)$ by replacing every edge with a digon. The graph $G'(P)$ is therefore Eulerian, and its Eulerian walks (i.e., walks that traverse every edge of $G(P)$ precisely twice), called *double traces* of $G(P)$, play a key role in modeling the construction process. Note that the argument shows that every graph admits a double trace.

Double traces with additional properties related to stability of the constructed polyhedra were introduced as a combinatorial model underlying these approaches to polypeptide polyhedra design in [15] and [8]. Stability of the resulting polyhedron depends on two additional properties: one relates to whether in the double trace the neighborhoods of vertices can be split, and the other defines whether the edges of the double trace are traversed twice in the same or in different directions.

To define the first property, let an alternate sequence $W = w_0 e_1 w_1 \dots w_{2m-1} e_{2m} w_{2m}$, where e_i is an edge between vertices w_{i-1} and w_i , be a double trace — a closed walk which traverses every edge of graph exactly twice. Note that we always consider vertex sequence of a double trace with indices taken modulo $2m$. (Since the graph $G(P)$ is simple, so are all our other graphs, except $G'(P)$. Hence, a double trace is completely described by listing the vertices of the corresponding walk and we sometimes write double trace as a sequence consisting only from vertices.) For a set of vertices $N \subseteq N(v)$, a double trace W has a *N -repetition* at vertex v (nontrivial N -repetition in [8]), if N is nonempty, $N \neq N(v)$, and whenever W comes to v from a vertex in N it also continues to a vertex in N . More formally W has a N repetition at v if the following implication holds:

$$\text{for every } i \in \{0, \dots, \ell - 1\}: \text{ if } v = w_i \text{ then } w_{i+1} \in N \text{ if and only if } w_{i-1} \in N.$$

Then, W is a *strong trace* if W is for every vertex v without N -repetitions at v . It is a nontrivial result of [8] that every graph admits a strong trace. A weaker concept of *d -stable trace* requires that whenever W has an N -repetition at some vertex v , then $|N| > d$. Fijavž et al. showed that G admits a d -stable trace if and only if $\delta(G) \geq d$ [8].

For the second property, note that there are precisely two directions to traverse an edge $e = uv$. If the same direction is used both times W traverses e , then e is a *parallel* edge w.r.t. W , otherwise it is an *antiparallel* edge. A double trace W is *parallel*, if all edges of G are parallel w.r.t. W and is *antiparallel*, if all the edges are antiparallel. Interestingly, antiparallel traces appeared (under a different name) two centuries ago in a study of properties of labyrinths by Tarry [21], who observed (in our language) that every connected graph admits an antiparallel double trace. Fijavž et al. extended this by characterizing the graphs that admit an antiparallel strong trace [8], and Rus upgraded the result to characterize graphs that admit an antiparallel d -stable trace [19]. The former characterization can be algorithmically verified using algorithms of [11], but regarding the latter, it is only known that the existence of antiparallel 1-stable traces can be verified using Thomassen’s modification of the aforementioned algorithm, as published in [22] and later corrected by Benevánt López and Soler Fernández in [2]. Similar modification of algorithm for spanning tree parity problem presented in [12] would work for $d > 1$ as well, rendering the problem “Does there exist an antiparallel d -stable trace in G ?” polynomially tractable. Some additional research was also made in [3] and [6].

It is easy to obtain new traces from a given trace: one can change direction of tracing or start at a different vertex. Also, if graph possesses certain symmetries, these may reflect in the trace. Such changes do not alter any properties of the trace, hence we call the resulting traces equivalent, and we are interested in non-equivalent traces, as introduced in [15]:

Definition 1.1 *Two double traces W and W' are called equivalent if W' can be obtained from W (i) by reversion of W , (ii) by shifting W , (iii) by applying a permutation on W induced by an automorphism of G , or (iv) using any combination of the previous three operations. If that is not the case, W and W' are non-equivalent.*

Two double traces W and W' are called *different* if their vertex sequences are not the same. Two different double traces may be equivalent. It is easy to see that equivalence of double traces is an equivalence relation on the set \mathcal{T} of all different double traces, and hence on any subset (such as strong traces, d -stable traces etc.). The main contribution of our paper is designing for each of the subsets of interest an algorithm that, for a given graph as an input, outputs precisely one representative of each equivalence class. This representative will be the unique minimal element for the following linear ordering, called *lexicographical ordering* of double traces. We assume that the vertices of G are linearly ordered as $v_0 < v_1 < \dots < v_{n-1}$, and that v_0, v_1 are adjacent. This linear ordering induces an ordering on the set of double traces as follows:

Definition 1.2 *Given two double traces $W = w_0 \dots w_{2m}$ and $W' = w'_0 \dots w'_{2m}$, W is lexicographically smaller or equal to W' , denoted $W \leq_{lex} W'$, if and only if $W = W'$ or the first w_i , which is different from w'_i , is smaller than w'_i .*

As lexicographical order is a linear order, it is clear that any finite set S of double traces has a unique lexicographically smallest member. We call that member the *canonical representative* of S .

For a more detailed treatment of double-trace related definitions we refer the reader to [8]. For other terms and concepts from graph theory not defined here, we refer to [23].

Let the *automorphism group* $\text{Aut}(G)$ of G be denoted by A . An automorphism $\pi \in A$ acts on \mathcal{T} by mapping a double trace $W = w_0 \dots w_{2m}$ to $\pi(W) = \pi(w_0) \dots \pi(w_{2m})$. Let $\rho : \mathcal{T} \rightarrow \mathcal{T}$ be a reversal that maps $W = w_0 \dots w_{2m}$ to $W' = w_0 w_{2m} \dots w_1$, and, for $i = 0, \dots, 2m$, let σ_i be an i -shift that maps $W = w_0 \dots w_{2m}$ to $W'' = w_i \dots w_{2m+i}$. Note that $\sigma_0 = \sigma_{2m} = \text{id}$. Then the group A , the group $R = \{\text{id}, \rho\}$, and the group $S = \{\sigma_i \mid i = 0, \dots, 2m - 1\}$ are three groups acting on \mathcal{T} (or any of its subsets). Note that groups R and S do not commute and $\langle R, S \rangle$ is a dihedral group of symmetries of a regular $(2m)$ -gon, where $E(G) = 2m$. Therefore the orbits of the direct product $\Gamma = A \times \langle R, S \rangle$ are precisely the equivalence classes of double traces for the relation from Definition 1.1. Hence, a canonical representative of each equivalence class is the lexicographically smallest element of each class. We say that a double trace is *canonical*, if it is the lexicographically smallest element of its orbit, meaning that every element of Γ maps it to a lexicographically larger (or equal) element. Note that to verify canonicity of a particular double trace, it is not enough to check whether the generators of Γ map it to a larger element (we leave finding an example to the reader).

It is easy to see that every canonical double trace starts with $v_0 v_1$ (by assumption, these two vertices are adjacent) and that every double trace is equivalent to at least one canonical double trace. Double traces (not necessary canonical) starting with $v_0 v_1$ are called *simple*. More details on graph automorphisms can be found in [13], but we do conclude this introduction with an example of the action of Γ on \mathcal{T} in the case of the tetrahedron.

In Figure 1 we graphically present the action of Γ on \mathcal{T} in the case of the tetrahedron. The vertices of a graph on each subfigure represent all 672 different strong traces of tetrahedron (generated with simple backtracking without eliminating the non-canonical traces). Two vertices t_1 and t_2 are then adjacent if they lie in the same orbit of Γ . Note that Γ partitions \mathcal{T} into 3 orbits of orders 288, 288, and 96. This fact coincides with the results of Table 1. Subgroups A , R , and S partition \mathcal{T} into 28 orbits of order 24, 336 orbits of order 2, and 56 orbits of order 12, respectively.

This is (to our knowledge) the first analyze of the automorphism group of a double trace. We proceed as follows. In Section 2, we use the automorphism group to devise a branch-and-bound algorithm that outputs each canonical strong double trace of G precisely once. The main idea of the algorithm is avoiding isomorphs by extending minimal forms. Such an idea was first presented in [18] where it was called the orderly generation. It is not difficult to see that with minor adjustments, this algorithm can enumerate other varieties of double traces, such as d -stable traces, parallel double traces, or antiparallel double traces. We conclude, in Section 3, with some numerical results that reveal possible varieties in designing polyhedral polypeptide nanostructures.

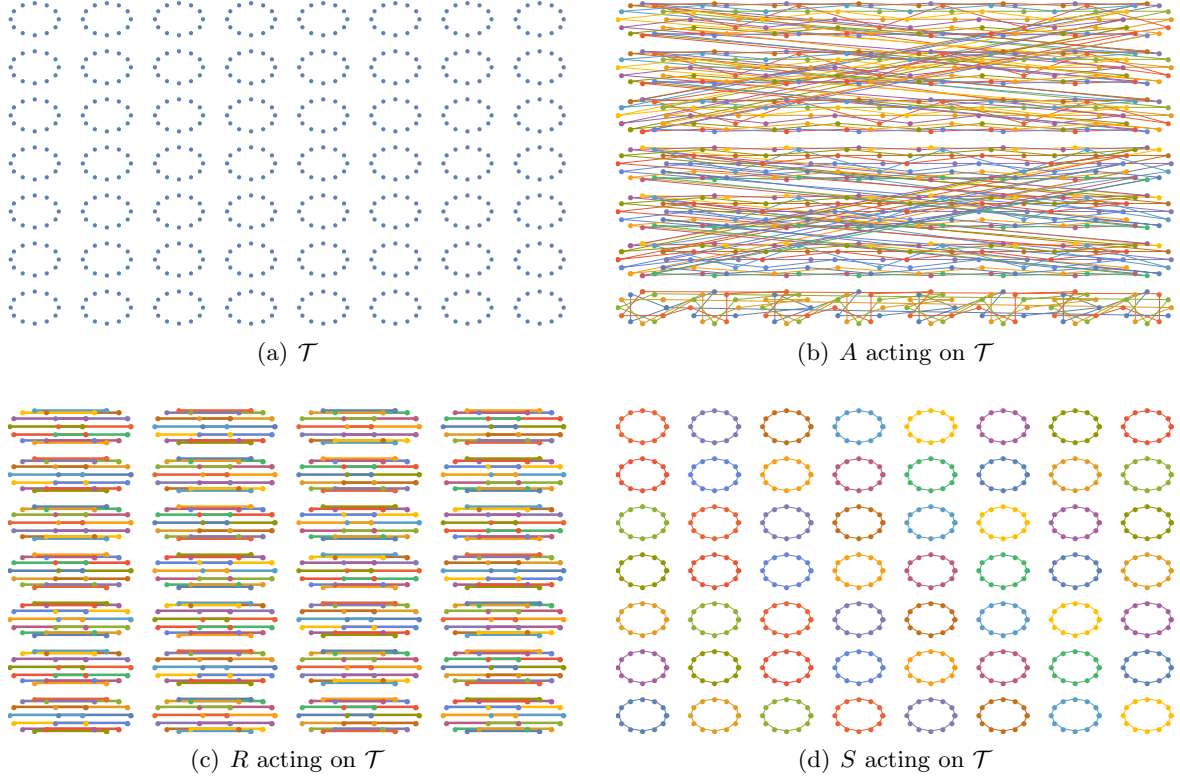


Figure 1: Graphical presentation of A , R , and S acting on the set \mathcal{T} of all 672 strong traces of a tetrahedron. Strong traces are presented as vertices of a graph (which consequently has 672 vertices), two being adjacent when at least one element of A or R or S map one into another. To make the presentation a bit more transparent some edges are left out at figures (b) and (d). Figure (b) shows 28 instances of C_{24} which should be replaced with 28 instances of K_{24} , while figure (d) shows 56 instances of C_{12} which should be replaced with 56 instances of K_{12} .

2 Enumerating strong traces with branch-and-bound strategy

In this section we assume that the n vertices of some arbitrary, but fixed, connected graph G with m edges are linearly ordered as $v_0 < v_1 < \dots < v_{n-1}$, and that v_0, v_1 are adjacent. Therefore every canonical double trace of G starts with v_0v_1 . We denote the automorphism group of double traces in graph G with Γ . To make the arguments more transparent, let W and W' from now on be two different double traces. We first give some additional observations.

Definition 2.1 Let $W = w_0 \dots w_{2m}$ be a double trace of a graph G . An initial segment $\text{init}(W)$ of W is the shortest continuous subsequence of W such that $\text{init}(W)$ starts in w_0 and contains all the vertices from $V(G)$.

Definition 2.2 Let $W = w_0 \dots w_{2m}$ be a double trace. Then an i -initial segment of W , denoted W_i , is a subsequence of first i vertices in W , i.e., $W_i = w_0 \dots w_{i-1}$.

Definition 2.3 A double trace W is i -canonical if and only if for every $\pi \in \Gamma$, the relation $W_i \leq_{\text{lex}} \pi(W_i)$ holds.

Lemma 2.4 If a double trace W of length $2m$ is canonical, then it follows that W is i -canonical for all i , $1 \leq i \leq 2m$.

Proof. Let W be a canonical double trace of length $2m$. Suppose that for some i , $1 \leq i < 2m$, W is not i -canonical. Then there exists $\pi \in \Gamma$, such that $\pi(W_i) <_{\text{lex}} W_i$. Because W is canonical, it follows that $W \leq_{\text{lex}} \gamma(W)$ for every $\gamma \in \Gamma$. Therefore, $W \leq_{\text{lex}} \pi(W)$. By Definition 1.2, it follows that at the first index j , where $w_j \neq \pi(w_j)$, $w_j < \pi(w_j)$. For every $i < j$, $W_i = \pi(W_i)$, while for every every $i \geq j$, W_i contains w_j and therefore $W_i <_{\text{lex}} \pi(W_i)$, a contradiction. \square

We first explain the auxiliary algorithms used in the main Algorithm 4. If G is a graph with m edges and $p \leq 2m$, then vertex sequence $W_p = w_0 \dots w_{p-1}$ is a *partial double trace* if there exists a double trace W of G for which W_p is its p -initial segment. Analogously we define partial double trace for other varieties of double traces (strong and d -stable traces). Let W_p be a partial double trace of length p . Set \mathcal{W} represent all double traces in G for which their p -initial segment is equal to W_p . While we say that W_p is lexicographically smaller than different partial double trace W'_p if $W_p = W'_p$ or the first w_i , which is different from w'_i , is smaller than w'_i , we say that W_p is canonical if at least one the double trace from \mathcal{W} is canonical. Stabilizer of a partial double trace W_p is defined as subset of all automorphisms in Γ which map at least one double trace from \mathcal{W} back to (not necessary the same) double trace from \mathcal{W} . *Feasible neighbors* of w_{p-1} in a partial double trace W_p is a subset of its neighbors $N(w_{p-1})$. For every feasible neighbor v then $W_{p+1} = w_0 \dots w_{p-1}v$ obtained from W_p by adding v also W_{p+1} should be a partial double trace. Analogously for partial strong traces and d -stable traces where we have to be careful that v does not cause any new nontrivial repetition of excessive order. For antiparallel or parallel double traces we additionally forbid vertices causing parallel or antiparallel edges in partial double trace, respectively.

Algorithm 1 loops through all the feasible neighbors of the last vertex w_{p-1} in a partial double trace $W_p = w_0 \dots w_{p-1}$ and check which of them, if added to W_p (and obtaining partial double trace W_{p+1}), will maintain a canonical partial double trace. Partial double traces obtained in this procedure are added to queue Q .

At each step we use the automorphism group of double traces Γ in order to eliminate all partial double traces that would not lead to a construction of a canonical double

Algorithm 1 EXTEND FEASIBLY

Input: a partial double trace $W_p = w_0 \dots w_{p-1}$, $A \subseteq \Gamma$, a queue of Q partial double traces
 $V' = \text{FEASIBLE_NEIGHBORS}(w_{p-1})$
 $V'' = \text{CANONICAL_EXTENSION}(V', W_p, A)$
for $v \in V''$ **do**
 $W_{p+1} = w_0 \dots w_{p-1}, v$
 $A_v = \text{PRUNE}(A, W_{p+1})$
 if W_{p+1} is canonical partial double trace **then**
 append (W_{p+1}, A_v) to Q

trace. We achieve that by considering only the lexicographically smallest representative of each orbit of the automorphism group. Simultaneously, we would like to fix vertices that are already in a partial double trace, since we have already checked it for canonicity. Therefore, Algorithm 2 returns the automorphisms that are in the stabilizer of partial double trace W_{p+1} (in each step only the last position p has to be checked). Note that until double trace is not completed, we can not determine if automorphism lies in a stabilizer of partial double trace, for all automorphisms from Γ . Problem is in shifting, since we can not always determine how all first p places of shifted partial double trace look like. Therefore we do not discard such automorphisms at this point.

Algorithm 2 PRUNE

Input: set of automorphisms of double traces A , partial double trace W_p
Output: pruned set of automorphisms of double traces A'
 $A' = \emptyset$
for $\pi \in A$ **do**
 if π in stabilizer of partial double trace W_p or if it can not be determined if π is in a stabilizer of W_p **then**
 append π to A'
return A'

Algorithm 3 loops through all the feasible neighbors of the last vertex w_{p-1} last added to a partial double trace $W_p = w_0 \dots w_{p-1}$ and denoted with $V \subseteq N(w_{p-1})$. For every $v \in V$ it constructs new partial double trace $W_{p+1} = w_0 \dots w_{p-1}v$ and analyze orbits of $\text{Aut}(G) \cap A$ (no shifts and reverses are allowed, therefore each orbit contains even smaller number of partial double trace) acting on set of these new partial double traces. Then for every such orbit O algorithm select vertex $v \in V$ for which partial double trace $W_p = w_0, \dots, w_{p-1}$ is lexicographically smallest of partial double traces in O . Note that in practice algorithm should only check the position p since Algorithm 2 ensures that for every $\pi \in \text{Aut}(G) \cap A$ vertices $w_0 \dots w_{p-1}$ are fixed.

Algorithm 3 CANONICAL EXTENSION

Input: partial double trace $W_p = w_0, \dots, w_{p-1}$, set of feasible neighbors $V \subseteq N(w_{p-1})$, set of automorphisms $A \subseteq \Gamma$

Output: set $V' \subseteq V$ containing for each orbit O of $\text{Aut}(G) \cap A$ vertex v for which $W_{p+1} = w_0, \dots, w_{p-1}, v$ is lexicographically smallest partial double trace of O

if $A = \emptyset$ or $A = \{\text{id}\}$ **then**
 return V
 $V' = \emptyset$

for $v \in V$ **do**
 $V'' = \{v\}$
 $v' = v$
 for $\pi \in \text{Aut}(G) \cap A$ **do**
 append $\pi(v)$ to V''
 if $(w_0, \dots, w_{p-1}, \pi(v)) <_{\text{lex}} (w_0, \dots, w_{p-1}, v')$ **then**
 $v' = \pi(v)$
 append (v') to V'
 for $v'' \in V''$ **do**
 remove v'' from V

return V'

We now present the main Algorithm 4, which enumerates strong traces for an arbitrary graph.

Algorithm 4 ENUMERATE STRONG TRACES

Input: a graph G with m edges, automorphism group Γ of double traces of G

Output: a list of all non-equivalent double traces L

$W_1 = v_0 v_1$
 $A = \text{Aut}(G)$
 $A = \text{PRUNE}(A, W_1)$
 $Q = \{(W_1, A)\}$
while Q not empty **do**
 $(W, A) = \text{head of } Q$
 remove (W, A) from Q
 if $|W| = 2m$ **then**
 add W to L
 else
 EXTEND FEASIBLY(W, A, Q)

return L

In the rest of the section, we prove the correctness of Algorithm 4.

Theorem 2.5 *Let W be a double trace, which was given as an output of Algorithm 4. Then W is canonical.*

Proof. Let $W = w_0, \dots, w_{2m}$ be a double trace obtained as an output of Algorithm 4. Suppose that W is not canonical. Then there exists a double trace W' and $\pi \in \Gamma$, such that $W' = \pi(W)$ and $\pi(W) <_{lex} W$. Let i be the smallest integer such that $w'_i \neq w_i$. Then $w'_i < w_i$ and $w_j = w'_j = \pi(w_j)$, for $0 \leq j < i$. For every $1 \leq j < i$ automorphism π fixes edge $w_{j-1}w_j$: $w_{j-1}w_j = \pi(w_{j-1})\pi(w_j)$, hence π is contained in the stabilizer of W_j . Consequently Algorithm 4 (Algorithm 2 to be more precise) does not eliminate π while pruning. Selecting w_i in Algorithm 4 (Algorithm 1 to be more precise) was not optimal since w'_i would produce lexicographically smaller equivalent double trace. This contradicts the fact that Algorithm 1 for every orbit select lexicographically smallest feasible neighbor. \square

Theorem 2.6 *Let W be a canonical double trace. Then W is given as an output of Algorithm 4.*

Proof. Suppose the contrary. Let $W = w_0, \dots, w_{2m}$ be a canonical double trace which is not given as an output of Algorithm 4. By observations made in Section 1, W starts with v_0v_1 . There exists the largest integer i (at least 1 if no other) such that W_i is the i -initial segment of some canonical double trace which is an output of Algorithm 4. Let \mathcal{W} be the set of all (canonical) double traces which are given as an output of Algorithm 4 and have W_i as their i -initial segment. Let $V_{\mathcal{W}, i+1}$ be the set of vertices that lie at the $(i+1)$ -th position in (canonical) double traces from \mathcal{W} . It follows that $w_{i+1} \notin V_{\mathcal{W}, i+1}$. Since W is a double trace, w_{i+1} was in the Algorithm 4 (Algorithm 3 to be more precise) part of feasible neighbors of w_i for every double trace from \mathcal{W} . Since it was never added it follows that in the same orbit of $Aut(G) \subseteq \Gamma$ than W also lies another (lexicographically smaller) double trace $W' \in \mathcal{W}$. That contradicts the fact that W is canonical. \square

We presented an algorithm which enumerates all non-equivalent double traces of graph. To enumerate only strong traces all only d -stable traces of graph we just have to complement the definition of feasible neighbors. Da namesto dvojnih obhodov v splošnem, štejemo le stroge ali le d -stabilne obhode je potrebno le spremeniti, kaj so dopustni sosedi vozlišča v , ki smo ga na i -tem koraku dodali v dvojni obhod. Pri strogih obhodih moramo tako paziti, da se ne pojavi nobena netrivialna ponovitev, pri d -stabilnih obhodih pa, da se ne pojavi nobena ponovitev reda $\leq d$. Podobno lahko preštevamo tudi paralelne ali antiparalelne dvojne obhode. Za povezavo $e = uv$, ki je bila v dvojnem obhodu že prečkana, si je potrebno zapomniti ali smo jo prečkali v smeri od u proti v ali pa v obratni smeri. Glede na to (in desjto ali štejemo paralelne ali antiparalelne obhode) potem ustrezno popravimo dopustne sosede vozlišča u in v .

3 Concluding remarks and numerical results

We conclude with some numerical results. In Tables 1, 2, and 3 we present enumerations of non-equivalent strong traces for platonic solids, prisms, and some other interesting solids which could be the next candidates to be constructed from coiled-coil-forming segments. Note that d , n , m , ST , aST , and pST stand for the degree of the graph (if a graph is regular), the number of its vertices and edges, the number and the CPU time used to enumerate strong traces, the number and the CPU time used to enumerate antiparallel strong traces, and number and CPU time used to enumerate parallel strong traces in Tables 1, 2, and 3, respectively. Note that the listed CPU times are measured in seconds. In addition to the number of strong traces, the algorithm for every strong trace also returns its vertex sequence. Therefore it can be used for a thoroughly analysis of some properties that nanostructures self-assembled from these strong traces would have. Further, this analysis help to select a strong trace with the maximal probability to construct a stable nanostructure of desired shape.

graph	d	n	m	ST		pST	
				#	CPU time	#	CPU time
tetrahedron	3	4	6	3	0.005	0	-
cube	3	8	12	40	0.01	0	-
octahedron	4	6	12	21479	1.86	262	0.056
dodecahedron	3	20	30	2532008	2242.31	0	-

Table 1: Number of strong traces and parallel strong traces for platonic solids

graph	d	n	m	ST		aST	
				#	CPU time	#	CPU time
Y_3	3	6	9	25	0.007	2	0.005
Y_4	3	8	12	40	0.01	0	-
Y_5	3	10	15	634	0.066	10	0.006
Y_6	3	12	18	3604	0.377	0	-
Y_7	3	14	21	21925	3.51	76	0.024
Y_8	3	16	24	134008	32.5	0	-
Y_9	3	18	27	833685	233.7	536	0.430
Y_{10}	3	20	30	5212520	2280.06	0	-

Table 2: Number of strong traces and antiparallel strong traces for prisms

All the calculations were made using Algorithm 4 and computational resources at SageMathCloud [4]. It was observed in [8], that a graph G admits a parallel strong

graph	d	n	m	ST		aST	
				#	CPU time	#	CPU time
4-pyramid	-	5	8	52	0.004	4	0.008
3-bipyramid	-	5	9	470	0.013	0	-

Table 3: Number of strong traces and antiparallel strong traces in 4-pyramid and 3-bipyramid

trace if and only if G is Eulerian, and that G admits an antiparallel strong trace if and only if there exists a spanning tree T of G with the property that every component of the co-tree $G - E(T)$ is even. Therefore, we omit the information about antiparallel and parallel strong traces for graphs not admitting them. Some of these calculations were already presented in [14] and [15].

Another possible approach to strong trace construction exploits the observation that a strong trace can be nicely drawn on a surface in which the given graph is embedded. This surface can be cut along certain edges which results in one or more surfaces with boundary. Each of the resulting surfaces with boundary carries a part of the information about the strong trace. The strong trace can be reconstructed by gluing those smaller pieces back together. This topological approach will be elaborated in [1].

Acknowledgments

The authors would like to thank to Anders Skovgaard Knudsen who independently calculated the number of strong traces in platonic solids and shared the results for comparison.

This research was supported in part by Slovenian Research Agency under research grants L7-5554 and P1-0294.

References

- [1] N. Bašić, D. Bokal, T. Pisanski, J. Rus, *Graph embeddings yield natural strong trace realizations*, in preparation.
- [2] E. Benevante López, D. Soler Fernández, *Searching for a strong double tracing in a graph*, Sociedad de Estadística e Investigación Operativa Top Vol. 6 (1998), 123–138.
- [3] H. J. Broersma and F. Göbel, *k-Traversable graphs*, Ars Combin. **29** (1990), no. A, 141–153, Twelfth British Combinatorial Conference (Norwich, 1989).

- [4] The Sage Developers, *Sage Mathematics Software (Version 6.9.(2015-10-10))*, 2015, <http://www.sagemath.org>.
- [5] R. B. Eggleton and D. K. Skilton, *Double tracings of graphs*, Ars Combin. **17** (1984), no. A, 307–323.
- [6] J. A. Ellis-Monaghan, A. McDowell, I. Moffatt, and G. Pangborn, *DNA origami and the complexity of Eulerian circuits with turning costs*, Nat. Comput. **14** (2015), no. 3, 491–503.
- [7] L. Euler, *Solutio problematis ad geometriam situs pertinentis*, Commentarii Academiae Scientiarum Imperialis Petropolitanae **8** (1741), 128–140.
- [8] G. Fijavž, T. Pisanski, and J. Rus, *Strong traces model of self-assembly polypeptide structures*, MATCH Commun. Math. Comput. Chem. **71** (2014), no. 1, 199–212.
- [9] H. Fleischner, *Eulerian graphs and related topics. Part 1. Vol. 1*, Annals of Discrete Mathematics, vol. 45, North-Holland Publishing Co., Amsterdam, 1990.
- [10] H. Fleischner, *Eulerian graphs and related topics. Part 1. Vol. 2*, Annals of Discrete Mathematics, vol. 50, North-Holland Publishing Co., Amsterdam, 1991.
- [11] M. L. Furst, J. L. Gross, and L. A. McGeoch, *Finding a maximum-genus graph imbedding*, J. Assoc. Comput. Mach. **35** (1988), no. 3, 523–534.
- [12] H. N. Gabow, M. Stallman, *An Augmenting Path Algorithm for Linear Matroid Parity*, Combinatorica **6** 2 (1986), 123–150.
- [13] C. Godsil and G. Royle, *Algebraic Graph Theory*, Graduate Texts in Mathematics, vol. 207, Springer-Verlag, New York, 2001.
- [14] H. Gradišar, S. Božič, T. Doles, D. Vengust, I. Hafner Bratkovič, A. Mertelj, B. Webb, A. Šali, S. Klavžar, and R. Jerala, *Design of a single-chain polypeptide tetrahedron assembled from coiled-coil segments*, Nat. Chem. Biol. **9** (2013), no. 6, 362–366.
- [15] S. Klavžar and J. Rus, *Stable traces as a model for self-assembly of polypeptide nanoscale polyhedrons*, MATCH Commun. Math. Comput. Chem. **70** (2013), no. 1, 317–330.
- [16] V. Kočar, S. Božič Abram, T. Doles, N. Bašić, H. Gradišar, T. Pisanski, and R. Jerala, *Topofold, the designed modular biomolecular folds: polypeptide-based molecular origami nanostructures following the footsteps of dna*, WIREs Nanomed. Nanobiotechnol. **7** (2015), no. 2, 218–237.

- [17] T. Pisanski and A. Žitnik, *Representing graphs and maps*, Topics in topological graph theory, Encyclopedia Math. Appl., vol. 128, Cambridge Univ. Press, Cambridge, 2009, pp. 151–180.
- [18] R. C. Read, *Every one a winner*, Annals Discrete Math. 2 (1978), 107–120.
- [19] J. Rus, *Antiparallel d -stable traces and a stronger version of Ore problem*, (2015), submitted.
- [20] G. Sabidussi, *Tracing graphs without backtracking*, Methods of Operations Research XXV, Part 1 (R. Henn, P. Kall, B. Korte, O. Krafft, W. Oettli, K. Ritter, J. Rosenmüller, N. Schmitz, H. Schubert, and W. Vogel, eds.), First Symposium on Operations Research, University of Heidelberg, Verlag Anton Hain, June 1977, pp. 314–332.
- [21] G. Tarry, *Le problème des labyrinthes*, Nouv. Ann. **3** (1895), no. XIV, 187–190.
- [22] C. Thomassen, *Bidirectional retracting-free double tracings and upper embeddability of graphs*, J. Combin. Theory Ser. B **50** (1990), 198–207.
- [23] D. B. West, *Introduction to Graph Theory*, Prentice Hall, Upper Saddle River, 1996.